On the Sizes of DPDAs, PDAs, LBAs

by Richard Beigel and William Gasarch

Abstract

There are languages A such that there is a Pushdown Automata (PDA) that recognizes A which is much smaller than any Deterministic Pushdown Automata (DPDA) that recognizes A. There are languages A such that there is a Linear Bounded Automata (Linear Space Turing Machine, henceforth LBA) that recognizes A which is much smaller than any PDA that recognizes A. There are languages A such that both A and $\overline{A}$ are recognizable by a PDA, but the PDA for A is much smaller than the PDA for $\overline{A}$. There are languages A_1, A_2 such that $A_1, A_2, A_1 \cap A_2$ are recognizable by a PDA, but the PDA for A_1 and A_2 are much smaller than the PDA for $A_1 \cap A_2$. We investigate these phenomena and show that, in all these cases, the size difference is captured by a function whose Turing degree is on the second level of the arithmetic hierarchy.

Our theorems lead to infinitely-many-n results. For example: for-infinitely-many-n there exists a language A_n recognized by a DPDA such that there is a small PDA for A_n, but any DPDA for A_n is very large. We look at cases where we can get all-but-a-finite-number-of-n results, though with much smaller size differences.

Keywords: Pushdown Automata; Context Free Languages; Linear Bounded Automata; length of description of languages

1 Introduction

Let DPDA be the set of Deterministic Push Down Automata, PDA be the set of Push Down Automata, and LBA be the set of Linear Bounded Automata (usually called nondeterministic linear-space bounded Turing Machines). Let $L(\mathrm{DPDA})$ be the set of languages recognized by DPDAs (similar for $L(\mathrm{PDA})$ and $L(\mathrm{LBA})$). It is well known that

$$L(\text{DPDA}) \subset L(\text{PDA}) \subset L(\text{LBA}).$$

Our concern is with the *size* of the DPDA, PDA, LBA. For example, let $A \in L(\text{DPDA})$. Is it possible that there is a PDA for A that is much smaller than any DPDA for A? For all adjacent pairs above we will consider these questions. There have been related results by Valiant [16], Schmidt [12], Meyer and Fischer [11], Hartmanis [6], and Hay [7]. We give more details on their results later.

Throughout the paper Σ is a finite alphabet and \$ is a symbol that is not in Σ. All of our languages will either be subsets of Σ^* or $(\Sigma \cup \{\$\})^*$.

Convention 1.1 A *device* will either be a recognizer (e.g., a DFA) or a generator (e.g., a regular expression). We will use $\mathcal{M}$ to denote a set of devices (e.g., DFAs). We will refer to an element of $\mathcal{M}$ as an $\mathcal{M}$-device. If P is an $\mathcal{M}$-device then let $L(P)$ be the language recognized or generated by P. Let $L(\mathcal{M}) = \{L(P) : P \in \mathcal{M}\}$.

Def 1.2 Let $\mathcal{M}$ and $\mathcal{M}'$ be two sets of devices such that $L(\mathcal{M}) \subseteq L(\mathcal{M}')$. (e.g., DFAs and DPDAs). A *bounding function for* $(\mathcal{M}, \mathcal{M}')$ is a function f such that for all $A \in L(\mathcal{M})$, if $A \in L(\mathcal{M}')$ via a device of size n then $A \in L(\mathcal{M})$ via a device of size $\le f(n)$.

Def 1.3

1. The *size of a DFA or NDFA* is its number of states.
2. The *size of a DPDA or PDA* is the sum of its number of states and its number of symbols in the stack alphabet.
3. The *size of a CFG or CSL* is its number of nonterminals.

4. The *size of an LBA* is the sum of its number of states and its number of symbols in the alphabet (note that the alphabet used by the Turing machine may be bigger than the input alphabet).

We now give some examples and known results.

Example 1.4 Known Upper Bounds:

1. $f(n) = 2^n$ is a bounding function for (DFA,NDFA) by the standard proof that $L(\mathrm{NDFA}) \subseteq L(\mathrm{DFA})$.

2. $f(n) = n^{n^{n^{O(n)}}}$ is a bounding function for (DFA,DPDA). This is a sophisticated construction by Stearns [14].

3. $f(n) = 2^{2^{O(n)}}$ is a bounding function for (DFA,DPDA). This is a sophisticated construction by Valiant [18]. Note that this is a strict improvement over the construction of Stearns.

4. $f(n) = O(n^{O(1)})$ is a bounding function for (CFG,PDA). This can be obtained by an inspection of the proof that $L(\mathrm{PDA}) \subseteq L(\mathrm{CFG})$.

5. $f(n) = O(n)$ is a bounding function for (PDA,CFG). This can be obtained by an inspection of the proof that $L(\mathrm{CFG}) \subseteq L(\mathrm{PDA})$.

6. $f(n) = O(n)$ is a bounding function for (CSG,LBA). This can be obtained by an inspection of the proof that $L(\mathrm{LBA}) \subseteq L(\mathrm{CSG})$.

7. $f(n) = O(n)$ is a bounding function for (LBA,CSG). This can be obtained by an inspection of the proof that $L(\mathrm{CSG}) \subseteq L(\mathrm{LBA})$.

Example 1.5 Known Lower Bounds:

1. Meyer and Fischer [11] proved that (1) If f is the bounding function for (DFA,NDFA) then $2^n \le f(n)$. (2) If f is the bounding function for (DFA,DPDA) then $2^{2^{O(n)}} \le f(n)$. (3) If f is the bounding function for (DFA,CFG) then $HALT \le_{\mathrm{T}} f$. The sets they used for (3) were finite.

2. Let UCFG be the set all unambiguous context free grammars. Valiant [16] showed that if f is the bounding function for (DPDA,UCFG) then $HALT \le_{\mathrm{T}} f$.

3. Schmidt [12] showed that if f is the bounding function for (UCFG,CFG) then $HALT \le_{\mathrm{T}} f$.

4. Hartmanis [6] showed that if f is the bounding function for (DPDA,PDA) then $HALT \le_{\mathrm{T}} f$.

5. Harel and Hirst ([5], Proposition 14 and Corollary 15 of the journal version, (Proposition 12 and Corollary 13 of the conference version) have shown that if g is computable then g is not a bounding function for (DPDA,PDA) in a very strong way. They showed that *for all* $n > 0$ there is a language L_n such that (1) there is a PDA for L_n of size $O(n)$, but (2) any DPDA for L_n requires size at least $g(n)$.

6. Hay [7] showed that if f is the bounding function for (DPDA,PDA) then $HALT <_{\mathrm{T}} f$. She also showed that there is a bounding function f for (DPDA,PDA) such that $f \le_{\mathrm{T}} INF$. (INF is the set of all indices of Turing machines that halt on an infinite number of inputs. It is complete for the second level of the arithmetic hierarchy and hence strictly harder than $HALT$.)

7. Gruber et al. [4] proved several general theorems about sizes of languages. Hay's result above is a corollary of their theorem.

Note 1.6 The results above that conclude $HALT \le_{\mathrm{T}} f$ were not stated that way in the original papers. They were stated as either f *is not recursive* or f *is not recursively bounded*. However, an

inspection of their proofs yields that they actually proved $HALT \leq_{\mathrm{T}} f$.

Def 1.7 Let $\mathcal{M}$ be a set of devices. A *c-bounding function for* $\mathcal{M}$ is a function f such that for all A that are recognized by an $\mathcal{M}$-device of size n, if $\overline{A} \in L(\mathcal{M})$ then it is recognized by an $\mathcal{M}$-device, of size $\leq f(n)$. One linguistic issue— we will write (for example) *c-bounding function for PDAs* rather than *c-bounding function for PDA* since the former flows better verbally.

We now give some examples and known results.

Example 1.8

1. $f(n) = 2^n$ is a c-bounding function for NDFAs. This uses the standard proofs that $L(\mathrm{NDFA}) \subseteq L(\mathrm{DFA})$ and that $L(\mathrm{DFA})$ is closed under complementation.

2. $f(n) = O(n)$ is a c-bounding function for DPDAs. This is an easy exercise in formal language theory.

3. $f(n) = O(n)$ is a c-bounding function for LBAs. This can be obtained by an inspection of the proof, by Immerman-Szelepcsenyi [8, 15], that nondeterministic linear space is closed under complementation.

Def 1.9 Let $\mathcal{M}$ be a set of devices. An *i-bounding function for* $\mathcal{M}$ is a function f such that for all A_1, A_2 that are recognized by an $\mathcal{M}$-device of size n, if $A_1 \cap A_2 \in L(\mathcal{M})$ then it is recognized by an $\mathcal{M}$- device, of size $\leq f(n)$. One linguistic issue— we will write (for example) *i-bounding function for PDAs* rather than *i-bounding function for PDA* since the former flows better verbally.

Example 1.10

1. $f(n) = 2n$ is an i-bounding function for DFA. This uses the standard proofs that $L(\mathrm{DFA})$ is closed under intersection.

2. $f(n) = 2^{2n}$ is an i-bounding function for NDFAs. Convert both NDFAs to DFAs and then use the standard proof that $L(\mathrm{DFA})$ is closed under intersection.

Note 1.11 We will state our results in terms of DPDAs, PDAs, and LBAs. Hence you may read expressions like $L(\mathrm{PDA})$ and think *isn't that just CFLs?* It is. We do this to cut down on the number of terms this paper refers to.

2 Facts and Notation

We will need the following notation and facts to state our results. We will prove the last item since it seems to not be as well known as the others.

Fact 2.1

1. $M_0, M_1, M_2, \ldots$ *is a standard numbering of all deterministic Turing Machines.*

2. $M_{e,s}(x)$ *is the result of running* $M_e(x)$ *for* s *steps.*

3. $HALT$ *is the set* $\{(e,x) : (\exists s)[M_{e,s}(x)$ *halts* $]\}$. $HALT$ *is* Σ_1*-complete. Hence any* $\exists$ *question can be phrased as a query to* $HALT$. *Note that any* $(\forall)$ *question can also be phrased as a query; however, you will have to negate the answer.*

4. INF *is the set* $\{e : (\forall x)(\exists y, s)[y > x \wedge M_{e,s}(y)$ *halts* $]\}$. INF *is* Π_2*-complete. Hence any* $(\forall)(\exists)$ *question can be phrased as a query to* INF. *Note that any* $(\exists)(\forall)$ *question can also be phrased as a query; however, you will have to negate the answer.*

5. $A \leq_{\mathrm{T}} B$ *means that* A *is decidable given complete access to set* B. *This can be defined formally with oracle Turing machines.*

6. $f \leq_{\mathrm{T}} HALT$ *iff there exists a computable* g *such that, for all* n, $f(n) = \lim_{s\to\infty} g(n,s)$. *We can take* g *to have complexity* $O(\log(n+s))$*, or even lower. This result is due to Shoenfield [13] and is referred to as The Shoenfield Limit Lemma. It is in most computability*

theory books. Note that the domain f is N*, the domain of g is* $\mathsf{N} \times \mathsf{N}$*, and the codomain of both f and g is* N*. Hence the* $\lim_{s\to\infty} g(n,s)$ *means that* $(\exists s_0, x)(\forall s \ge s_0)[g(n,s) = x]$.

Proof: We just prove Part 6.

Let $HALT_s = \{(e,x) : [M_{e,s}(x)$ halts $\}$.

Assume $f \le_{\mathrm{T}} HALT$ via oracle Turing machine $M_i^{()}$. Hence, for all n, $M_i^{HALT}(n)$ halts and is equal to $f(n)$. Since $M^{HALT}(n)$ halts it does so in a finite amount of time and using only a finite number of queries to $HALT$. Hence there exists s such that for all $t \ge s$, $f(n) = M_{i,t}^{HALT_t}(n)$. Therefore the following computable function g works.

$$g(n,s) = \begin{cases} 0 & \text{if } M_{i,s}^{HALT_s}(n) \text{ has not converged} \\ M_{i,s}^{HALT_s}(n) & \text{if it has converged} \end{cases} \tag{1}$$

We can obtain a g of very low complexity by (for example) having $g(n,s)$ computer $M_{i,\lg^* s}^{HALT_{\lg^* s}}$ instead of $M_{i,s}^{HALT_s}$

Assume there exists a computable g such that, for all n, $f(n) = \lim_{s\to\infty} g(n,s)$. The following algorithm shows $f \le_{\mathrm{T}} HALT$.

ALGORITHM

Input(n)

For $s_0 = 1$ to ∞ (we will show that this terminates)

Ask $HALT$ $(\forall s \ge s_0)[g(n,s) = g(n,s+1)]$.

If YES then output $g(n,s)$ and STOP.

END OF ALGORITHM

Since $\lim_{s\to\infty} g(n,s)$ exists there will be an s_0 such that the answer to the $HALT$ question is YES. Hence the algorithm terminates.

$\blacksquare$

3 Summary of Results

In this section we summarize our results. We also present them in a table at the end of this section.

The results of Hartmanis [6] and Hay [7] mentioned in Exercise 1.5 above leave open the exact Turing degree of the bounding function for (DPDA,PDA). In Section 4 we resolve this question by proving a general theorem from which we obtain the following:

1. If f is a bounding function for (DPDA,PDA) then $INF \leq_{\rm T} f$.

2. There exists a bounding function for (DPDA,PDA) such that $f \leq_{\rm T} INF$. (Hay [7] essentially proved this; however, we restate and reprove in our terms.)

3. If $INF \not\leq_{\rm T} f$ then for infinitely many n there exists a language $A_n \in L(\mathrm{DPDA})$ such that (1) any DPDA that recognizes A_n requires size $\geq f(n)$, (2) there is a PDA of size $\leq n$ that recognizes A_n. (This follows from Part 1.)

4. If f is a bounding function for (PDA,LBA) then $INF \leq_{\rm T} f$.

5. There exists a bounding function for (PDA,LBA) such that $f \leq_{\rm T} INF$.

6. If $INF \not\leq_{\rm T} f$ then for infinitely many n there exists a language $A_n \in L(\mathrm{PDA})$ such that (1) any PDA that recognizes A_n requires size $\geq f(n)$, (2) there is an LBA of size $\leq n$ that recognizes A_n. (This follows from Part 4.)

In Section 5 and 6 we find the exact Turing degree of the c-bounding function and the i-bounding function for PDAs. We obtain the following:

1. If f is a c-bounding function for PDAs then $INF \leq_{\rm T} f$.

2. There exists a c-bounding function for PDAs such that $f \leq_{\rm T} INF$.

3. If $INF \not\leq_{\mathrm{T}} f$ then for infinitely many n there exists a language A_n such that (1) $A_n, \overline{A_n} \in L(\mathrm{PDA})$, (2) there is no PDA of size $\leq f(n)$ for $\overline{A_n}$, but (3) there is a PDA of size $\leq n$ for A_n. (This follows from Part 1.)

4. Results 1,2,3 but with i-bounding functions instead of c-bounding functions.

Note that we have several results of the form *for infinitely many* $n \ldots$ that use unnatural languages. We would like to have *for all but finitely many* $n \ldots$ results that involve natural languages. We need the following definitions.

Def 3.1 Let $A(n)$ be a statement about the natural number n. $A(n)$ *is true for almost all* n means that $A(n)$ is true for all but a finite number of n.

Def 3.2 (Informal) A language is *unnatural* if it exists for the sole point of proving a theorem.

Example 3.3

1. Languages that involve Turing configurations are not natural.

2. Languages created by diagonalization are not natural.

3. The language $\{ww : |w| = n\}$ is natural.

Note 3.4 We will sometimes state theorems as follows: *there exists a (natural) language such that* $\ldots$. If we do not state it that way then the language is unnatural.

In Sections 7 we obtain the following *for almost all* n results[1].

For almost all n there exists a (natural) language $A_n \in L(\mathrm{DPDA})$ such that

1. Any DPDA for A_n requires size $\geq 2^{2^{\Omega(n)}}$.

[1]Harel and Hirst[5] obtained these independently 21 years ago. We comment on how their proofs and our proofs differ in Section 7.

2. There is a PDA of size $O(n)$ that recognizes A_n.

Note 3.5 As noted in Example 1.5.5 Harel and Hirst [5] have a stronger result; however, the language they uses is not natural.

Section 7 also has the following result. In fact, we prove this result first and derive the result above from it.

For almost all n there exists a (natural) language A_n such that

1. $A_n, \overline{A_n} \in L(\mathrm{PDA})$.
2. Any PDA for $\overline{A_n}$ requires size $\geq 2^{2^{\Omega(n)}}$.
3. There is a PDA of size $O(n)$ that recognizes A_n.

In Section 8 we show the following:

For almost all n there exists a (natural) language $A_n \in L(\mathrm{PDA})$ such that

1. Any PDA for A_n requires size $\geq 2^{2^{\Omega(n)}}$.
2. There is an LBA of size $O(n)$ that recognizes A_n.

In Section 9 we obtain[2] a *for almost all* n result for (PDA,LBA):

Let f be any function such that $f \leq_{\mathrm{T}} HALT$. For almost all n there exists a finite language A_n such that

1. Any PDA for A_n requires size $\geq f(n)$.
2. There is an LBA of size $O(n)$ that recognizes A_n.

[2]Meyer originally claimed this result. See the discussion in Section 9.

We summarize our results in the following tables.

The first table: The first two columns, $\mathcal{M}$ and $\mathcal{M}'$, indicate that we are looking at the bounding functions f for $(\mathcal{M}, \mathcal{M}')$. The third column yields a property that any such f must have. The fourth column states whether we get as a corollary to the proof a result about for-infinitely-many n (io) or for-almost-all-n (ae). The fifth column states if the sets involved are natural or not. The sixth column states if the condition on f from the third column captures exactly the Turing degree of the bounding function. For example, in the first column we see that any bounding function f for $(\mathrm{DPDA}, \mathrm{PDA})$ has to satisfy $INF \leq_{\mathrm{T}} f$; however, the YES in column six indicates that there is a bounding function of this Turing degree.

$\mathcal{M}$	$\mathcal{M}'$	$(\forall f)$	io/ae	Nat?	Exact TD?
DPDA	PDA	$INF \leq_{\mathrm{T}} f$	io	NO	YES
PDA	LBA	$INF \leq_{\mathrm{T}} f$	io	NO	YES
DPDA	PDA	$f \geq 2^{2^{\Omega(n)}}$	ae	YES	NO
PDA	LBA	$f \geq 2^{2^{\Omega(n)}}$	ae	YES	NO
PDA	LBA	$f \not\leq_{\mathrm{T}} HALT$	ae	NO	YES

The second table: The first two columns are a set of devices, $\mathcal{M}$, and an operation Op (either Complementation (c) or Intersection (i)). We are concerned with the Op-bounding functions f for $\mathcal{M}$. The third column yields a property that any such f must have. The fourth column states whether we get as a corollary to the proof a result about for-infinitely-many n (io) or for-almost-all-n (ae). The fifth column states if the sets involved are natural or not. The sixth column states if the condition on f from the third column captures exactly the Turing degree of the bounding function.

$\mathcal{M}$	Op	$(\forall f)$	io/ae	Nat?	Exact TD?
PDA	Complementation	$INF \le_{\mathrm{T}} f$	io	NO	YES
PDA	Intersection	$INF \le_{\mathrm{T}} f$	io	NO	YES
PDA	Complementation	$f \ge 2^{2^{\Omega(n)}}$	ae	YES	NO

4 Bounding Functions for (DPDA,PDA) and (PDA,LBA)

In this section we prove a general theorem about bounding functions and then apply it to both (DPDA,PDA) and (PDA,LBA). In both cases we show that the Turing degree of the bounding function is in the second level of the arithmetic hierarchy.

We will need to deal just a bit with actual Turing Machines.

Def 4.1 Let M be a deterministic Turing Machine. A *configuration (henceforth config)* of M is a string of the form $\alpha{}^{q}_{\sigma}\beta$ where $\alpha, \beta \in \Sigma^*$, $\sigma \in \Sigma$, and $q \in Q$. We interpret this as saying that the machine has $\alpha\sigma\beta$ on the tape (with blanks to the left and right), is in state q, and the head is looking at the square where we put the ${}^{q}_{\sigma}$. Note that from the config one can determine if the machine has halted, and also, if not, what the next config is.

Notation 4.2 If C is a string then C^R is that string written backwards. For example, if $C = aaba$ then $C^R = abaa$.

Def 4.3 Let $e, x \in \mathsf{N}$. We assume that any halting computation of M_e takes an even number of steps.

1. Let $ACC_{e,x}$ be the set of all sequences of config's represented by

$$\$C_1\$C_2^R\$C_3\$C_4^R\$ \cdots \$C_s^R\$$$

   such that

- $|C_1| = |C_2| = \cdots = |C_s|$.
- The sequence $C_1, C_2, \ldots, C_s$ represents an accepting computation of $M_e(x)$.

2. Let $ACC_e = \bigcup_{x \in \mathsf{N}} ACC_{e,x}$.

Hartmanis [6] proved the following lemma.

Lemma 4.4 *For all e, x, $\overline{ACC_{e,x}} \in L(\mathrm{PDA})$. For all e, $\overline{ACC_e} \in L(\mathrm{PDA})$. In both cases it is computable to take the parameters ((e, x) or e) and obtain the PDA.*

Def 4.5 Let $\mathcal{M}$ and $\mathcal{M}'$ be two sets of devices.

1. $\mathcal{M} \subseteq \mathcal{M}'$ *effectively* if there is a computable function that will, given an $\mathcal{M}$-device P, output an $\mathcal{M}'$-device P' such that $L(P) = L(P')$.

2. $\mathcal{M}$ is *effectively closed under complementation* if there is a computable function that will, given an $\mathcal{M}$-device P, output an $\mathcal{M}$-device P' such that $L(P') = \overline{L(P)}$.

3. The *non-emptiness problem for $\mathcal{M}$* is the following: *given an $\mathcal{M}$-device P determine if $L(P) \neq \emptyset$.*

4. The *membership problem for $\mathcal{M}$* is: *given an $\mathcal{M}$-device P and $x \in \Sigma^*$ determine if $x \in L(P)$.*

5. $\mathcal{M}$ is *size-enumerable* if there exists a list of devices $P_1, P_2, \ldots$ such that
   - $\mathcal{M} = \{L(P_i) : i \in \mathsf{N}\}$,
   - $(\forall i)[|P_i| \leq |P_{i+1}|]$, and
   - the function from i to P_i is computable.

   Note that DFA, NDFA, DPDA, PDA, LBA are all size-enumerable, however UCFG is not.

Theorem 4.6 *Let $\mathcal{M}$ and $\mathcal{M}'$ be two sets of devices such that the following hold.*

- $L(\mathcal{M}) \subseteq L(\mathrm{PDA}) \subseteq L(\mathcal{M}')$ *effectively.*
- *At least one of $\mathcal{M}$, $\mathcal{M}'$ is effectively closed under complementation.*
- *The non-emptiness problem for $\mathcal{M}$ is decidable.*
- *The membership problems for $\mathcal{M}$ and $\mathcal{M}'$ are decidable.*
- *Every finite set is in $L(\mathcal{M})$.*
- *$\mathcal{M}$ is size-enumerable.*

Then

1. *If f is a bounding function for $(\mathcal{M}, \mathcal{M}')$ then $HALT \leq_{\mathrm{T}} f$.*
2. *If f is a bounding function for $(\mathcal{M}, \mathcal{M}')$ then $INF \leq_{\mathrm{T}} f$.*
3. *There exists a bounding function $f \leq_{\mathrm{T}} INF$ for $(\mathcal{M}, \mathcal{M}')$.*
4. *If $INF \not\leq_{\mathrm{T}} f$ then for infinitely many n there exists a language $A_n \in L(\mathcal{M})$ such that (1) any $\mathcal{M}$-device that recognizes A_n requires size $\geq f(n)$, (2) there is an $\mathcal{M}'$-device of size $\leq n$ that recognizes A. (This follows from Part 2 so we do not prove it.)*

Proof:

1) If f is a bounding function for $(\mathcal{M}, \mathcal{M}')$ then $HALT \leq_{\mathrm{T}} f$.

Note that

- If $M_e(x)$ halts then $ACC_{e,x}$ has one string, which is the accepting computation of $M_e(x)$.
- If $M_e(x)$ does not halt then $ACC_{e,x} = \emptyset$.
- Given e, x one can construct a PDA for $\overline{ACC_{e,x}}$ by Lemma 4.4.

We give the algorithm for $HALT \le_{\rm T} f$. There will be two cases in it depending on which of $\mathcal{M}$ or $\mathcal{M}'$ is effectively closed under complementation.

ALGORITHM FOR $HALT \le_{\rm T} f$

1. Input(e,x)
2. Construct the PDA P for $\overline{ACC_{e,x}}$. Obtain the device Q in $\mathcal{M}'$ that accepts $\overline{ACC_{e,x}}$.
3. **Case 1:** $\mathcal{M}$ is effectively closed under complementation. Compute $f(|Q|)$. Let $D_1,\ldots,D_t$ be all of the $\mathcal{M}$-devices of size $\le f(|Q|)$. Create the $\mathcal{M}$ devices for their complements, which we denote $E_1,\ldots,E_t$.

   Case 2: $\mathcal{M}'$ is effectively closed under complementation. Find an $\mathcal{M}'$- device R for $\overline{L(Q)} = ACC_{e,x}$. Compute $f(|R|)$. Let $E_1,\ldots,E_t$ be all of the $\mathcal{M}$-devices of size $\le f(|R|)$.

   Note that at the end of step 3, regardless of which case happened, we have a set of $\mathcal{M}$-devices $E_1,\ldots,E_t$ such that

   $(e,x) \in HALT$ iff

   $(\exists 1 \le i \le t)[L(E_i)$ is one string which represents an accepting computation of $M_e(x)]$.
4. For each $1 \le i \le t$ (1) determine if $L(E_i)=\emptyset$ (2) if $L(E_i)=\emptyset$ then let w_i be the empty string, and (3) if $L(E_i)\ne\emptyset$ then, in lexicographical order, test strings for membership in $L(E_i)$ until you find a string in $L(E_i)$ which we denote w_i. If $\{w_1,\ldots,w_s\}$ contains a string representing an accepting computation of $M_e(x)$ then output YES. If not then output NO.

END OF ALGORITHM

2) If f is a bounding function for $(\mathcal{M},\mathcal{M}')$ then $INF \le_{\rm T} f$.

Note that

- If $e \in INF$ then $ACC_e \notin L(\text{PDA})$ since ACC_e is infinite and every string in it begins with $\$C_1\$C_2^R\$C_3\$$ where $|C_1| = |C_2^R| = |C_3|$.

- If $e \notin INF$ then $ACC_e \in L(\text{PDA})$ since ACC_e is finite.

- Given e one can construct a PDA for $\overline{ACC_e}$ by Lemma 4.4.

We give the algorithm for $INF \leq_{\text{T}} f$. There will be two cases in it depending on which of $\mathcal{M}$ or $\mathcal{M}'$ is effectively closed under complementation.

In the algorithm below we freely use Fact 2.1.2 to phrase $(\exists)$-questions as queries to $HALT$, and Part 1 to answer queries to $HALT$ with calls to f.

ALGORITHM FOR $INF \leq_{\text{T}} f$

1. Input(e)

2. Construct the PDA P for $\overline{ACC_e}$. Obtain the device Q in $\mathcal{M}'$ that accepts $\overline{ACC_e}$.

3. There are two cases.

   Case 1: $\mathcal{M}$ is effectively closed under complementation. Compute $f(|Q|)$. Let $D_1, \ldots, D_t$ be all of the $\mathcal{M}$-devices of size $\leq f(|Q|)$. Create the $\mathcal{M}$ devices for their complements, which we denote $E_1, \ldots, E_t$.

   Case 2: $\mathcal{M}'$ is effectively closed under complementation. Find an $\mathcal{M}'$- device R for $\overline{L(Q)} = ACC_e$. Compute $f(|R|)$. Let $E_1, \ldots, E_t$ be all of the $\mathcal{M}$-devices of size $\leq f(|R|)$.

   Note that at the end of step 3, regardless of which case happened, we have a set of $\mathcal{M}$-devices $E_1, \ldots, E_t$ such that

   $e \in INF \implies ACC_e \notin L(\text{PDA}) \implies ACC_e \notin L(\mathcal{M}) \implies ACC_e \notin \{L(E_1), \ldots, L(E_t)\}$
   $\implies (\exists x_1, \ldots, x_t)(\forall 1 \leq i \leq t)[ACC_e(x_i) \neq E_i(x_i)]$.

$e \notin INF \implies ACC_e$ is finite $\implies ACC_e \in L(\mathcal{M}) \implies (\exists 1 \le i \le t)[L(E_i) = ACC_e]$
$\implies \neg(\exists x_1, \ldots, x_t)(\forall 1 \le i \le t)[ACC_e(x_i) \neq E_i(x_i)]$.

4. Ask $(\exists x_1, \ldots, x_t)(\forall 1 \le i \le t)[ACC_e(x_i) \neq E_i(x_i)]$. (Note that ACC_e is decidable so this is a $(\exists)$ question.) If YES then output YES. If NO then output NO.

3) There exists a bounding function $f \le_{\mathrm{T}} INF$ for $(\mathcal{M}, \mathcal{M}')$.

In the algorithm below we freely use Fact 2.1.3 to phrase $(\exists)(\forall)$-questions as queries to INF.

Algorithm for f

1. Input(n)

2. MAX=0.

3. For every $\mathcal{M}'$-device P of size $\le n$ do the following

   (a) Ask $(\exists \mathcal{M}$-device $D)(\forall x)[P(x) = D(x)]$?

   (b) If YES then for $i = 1, 2, 3, \ldots$ ask $(\exists \mathcal{M}$-device $D, |D| = i)(\forall x)[P(x) = D(x)]$? until the answer is YES.

   (c) Let i be the value of i when the last step stopped. Note that $(\exists D, |D| = i)(\forall x)[P(x) = D(x)]$. If $i > MAX$ then $MAX = i$.

4. Output MAX.

∎

Corollary 4.7

1. *If f is a bounding function for (DPDA,PDA) then $INF \le_{\mathrm{T}} f$.*

2. *There exists a bounding function for (DPDA,PDA) such that $f \le_{\mathrm{T}} INF$.*

3. *If $INF \not\leq_{\mathrm{T}} f$ then for infinitely many n there exists a language $A_n \in L(\mathrm{DPDA})$ such that (1) any DPDA that recognizes A_n requires size $\geq f(n)$ for A_n, but (2) there is a PDA of size $\leq n$ that recognizes A_n.*

4. *If f is a bounding function for (PDA,LBA) then $INF \leq_{\mathrm{T}} f$.*

5. *There exists a bounding function for (PDA,LBA) such that $f \leq_{\mathrm{T}} INF$.*

6. *If $INF \not\leq_{\mathrm{T}} f$ then for infinitely many n there exists a language $A_n \in L(\mathrm{PDA})$ such that (1) any PDA that recognizes A_n requires size $\geq f(n)$, (2) there is an LBA of size $\leq n$ that recognizes A_n.*

Proof: We can apply Theorem 4.6 to all the relevant pairs since all of the premises needed are either obvious or well known. ∎

Note 4.8 Since deterministic time classes are effectively closed under complementation we can also apply Theorem 4.6 to get a corollaries about any deterministic time class that contains $L(\mathrm{PDA})$. Let

$$\omega = \inf\{\alpha : \text{Two } n \times n \text{ Boolean matrices can be multiplied in time } O(n^\alpha)\}.$$

Le Gall [3] has the current best upper bound: $\omega < 2.3728639$. We abuse notation by letting, for all $\alpha > 0$, $\mathrm{DTIME}(n^\alpha)$ be the set of all deterministic Turing machines that run in time $O(n^\alpha)$. Valiant [17] showed that that, for all $\alpha > \omega$, $L(\mathrm{PDA}) \subseteq L(\mathrm{DTIME}(n^\alpha))$. If Boolean matrix multiplication really is in $\mathrm{DTIME}(n^\omega)$ then so is $L(\mathrm{PDA})$. (Lee [9] showed that if $L(\mathrm{PDA}) \subseteq \mathrm{DTIME}(n^{3-\epsilon})$ then $\omega \leq 3 - (\epsilon/3)$; therefore the problems of $L(\mathrm{PDA})$ recognition and matrix multiplication are closely linked.) Hence, for all $\alpha > \omega$ (and possibly for ω also) we can obtain a corollary about $\mathrm{DTIME}(n^\alpha)$ that is similar to Corollary 4.7.

5 c-Bounding Functions for PDAs

Theorem 5.1

1. *If f is a c-bounding function for PDAs then $HALT \le_{\mathrm{T}} f$.*

2. *If f is a c-bounding function for PDAs then $INF \le_{\mathrm{T}} f$.*

3. *There exists a c-bounding function $f \le_{\mathrm{T}} INF$ for PDAs. (This is almost identical to the proof of Theorem 4.6.3 so we do not prove it.)*

4. *If $INF \not\le_{\mathrm{T}} f$ then for infinitely many n there exists a language A_n such that (1) $A_n, \overline{A_n} \in L(\mathrm{PDA})$, (2) there is no PDA of size $\le f(n)$ for $\overline{A_n}$, but (3) there is a PDA of size $\le n$ for A_n. (This follows from Part 2 so we do not prove it.)*

Proof:

- $P_1, P_2, \ldots,$ is a size-enumeration of PDAs.

- f is a c-bounding function for PDAs.

- g (when on two variables) is the computable function such that $\overline{ACC_{e,x}}$ is recognized by PDA $P_{g(e,x)}$.

- g (when on one variable) is the computable function such that $\overline{ACC_e}$ is recognized by PDA $P_{g(e)}$.

1) Let $t = f(g(e,x))$.

$(e,x) \in HALT$ iff $(\exists 1 \le a \le t)[L(P_a)$ is an accepting computation of $M_e(x)]$.

Since both the non-emptiness problem and the membership problem for PDAs is decidable this condition can be checked.

2) Let $t = f(g(e))$.

$e \in INF \implies ACC_e \notin L(\mathrm{PDA}) \implies ACC_e \notin \{L(P_1), \ldots, L(P_t)\} \implies$

$(\exists x_1,\ldots,x_t)(\forall 1 \le i \le t)[P_i(x_i) \ne ACC_e(x_i)]$.

$e \notin INF \implies ACC_e$ is finite $\implies ACC_e \notin \{L(P_1),\ldots,L(P_t)\} \implies$
$\neg(\exists x_1,\ldots,x_t)(\forall 1 \le i \le t)[P_i(x_i) \ne ACC_e(x_i)]$.

We can now use $f \le_{\mathrm{T}} HALT$ to determine if $(\exists x_1,\ldots,x_t)(\forall 1 \le i \le t)[P_i(x_i) \ne ACC_e(x_i)]$. is true or not. ∎

6 i-Bounding Functions for PDAs

Def 6.1 We use the same conventions for Turing machines as in Definition 4.3. Let $e, x \in \mathsf{N}$.

1. $ODDACC_{e,x}$ be the set of all sequences of config's represented by

$$\$C_1\$C_2^R\$C_3\$C_4^R\$\cdots\$C_s^R\$$$

   such that

   - $|C_1| = |C_2|$ and $|C_3| = |C_4|$ and ... and $|C_{s-1}| = |C_s|$.
   - For all odd i, C_{i+1} is the next config after C_i. (We have no restriction on, say, how C_2 and C_3 relate. They could even be of different lengths.)
   - C_s represents an accepting config.

2. Let $ODDACC_e = \bigcup_{x \in \mathsf{N}} ODDACC_{e,x}$.

3. $EVENACC_{e,x}$ be the set of all sequences of config's represented by

$$\$C_1\$C_2^R\$C_3\$C_4^R\$\cdots\$C_s^R\$$$

   such that

   - $|C_2| = |C_3|$ and $|C_4| = |C_5|$ and ... and $|C_{s-2}| = |C_{s-1}|$.

- For all even i, C_{i+1} is the next config after C_i. (We have no restriction on, say, how C_3 and C_4 relate. They could even be of different lengths. We also have no restriction on C_1 except that it be a config.)

4. Let $EVENACC_e = \bigcup_{x \in \mathsf{N}} EVENACC_{e,x}$.

Note that

1. $(e,x) \in HALT$ iff $ODDACC_{e,x} \cap EVENACC_{e,x}$ contains only one string and that string is an accepting computation of $M_e(x)$.

2. $e \in INF$ iff $ODDACC_e \cap EVENACC_e \notin L(\mathrm{PDA})$.

Using these two facts you can prove the theorem below in a manner similar to the proof of Theorem 5.1.

Theorem 6.2

1. *If f is an i-bounding function for PDAs then $HALT \le_{\mathrm{T}} f$.*

2. *If f is an i-bounding function for PDAs then $INF \le_{\mathrm{T}} f$.*

3. *There exists an i-bounding function $f \le_{\mathrm{T}} INF$ for PDA.*

4. *If $INF \not\le_{\mathrm{T}} f$ then for infinitely many n there exists languages $A_{n,1}$ and $A_{n,2}$ such that (1) $A_{n,1}, A_{n_2} \in L(\mathrm{PDA})$, (2) there is no PDA of size $\le f(n)$ for $A_{n,1} \cap A_{n,2}$, but (3) there is a PDA of size $\le n$ for $A_{n,1} \cap A_{n,2}$.*

7 A Double-Exp For-Almost-All Result Via a Natural Language for (DPDA,PDA)

We show that for almost all n there is a (natural) language A_n such that A_n has a small PDA but $\overline{A_n}$ requires a large PDA. We then use this to show that for almost all n there is a language A_n that has a small PDA but requires a large DPDA. Neither of these results is new. Harel and Hirst [5] have

essentially proved everything in this section. We include this section because some of our proofs are different from theirs and because in most cases they do not explicitly state the theorems. After every statement and proof in this section we briefly discuss what they did. We denote their paper by HH.

Lemma 7.1 *Let X, Y, Z be nonterminals. Let Σ be a finite alphabet.*

1. *For all $n \geq 2$ there is a PDA of size $O(\log n)$ that generates $\{Y^n\}$.*

2. *For all $n \geq 2$ there is a PDA of size $O(\log n)$ that generates $\{a, b\}^n$.*

3. *For all $n \geq 2$ there is a PDA of size $O(\log n)$ that generates $\{Y^{\leq n}\}$.*

4. *For all $n \geq 2$ there is a PDA of size $O(\log n)$ that generates $\{a, b\}^{\leq n}$.*

5. *For all $n \geq 2$ there is a PDA of size $O(\log n)$ that generates $\{Y^{\geq n}\}$.*

6. *For all $n \geq 2$ there is a PDA of size $O(\log n)$ that generates $\{a, b\}^{\geq n}$.*

Proof:

1) We show that there is a CFG of size $\leq 2 \lg n$ that generates $\{Y^n\}$ by induction on n.

If $n = 2$ then the CFG for $\{YY\}$ is

$S \to YY$

which has $2 = 2 \lg 2$ nonterminals.

If $n = 3$ then the CFG for $\{YYY\}$ is

$S \rightarrow Y_1 Y$

$Y_1 \rightarrow YY$

which has $3 \le 2\lg 3$ nonterminals.

Assume that for all $m < n$ there is a CFG of size $\le 2\lg m$ for $\{Y^m\}$. We prove this for n.

- n is even. Let G' be the CFG for $\{Y^{n/2}\}$ with the start symbol replaced by S'. The CFG G for $\{Y^n\}$ is the union of G' and the one rule $S \rightarrow S'S'$. This CFG has one more nonterminal than G'. Hence the number of nonterminals in G is $\le 2\lg(n/2) + 1 \le 2\lg n$

- n is odd. Let G' be the CFG for $\{Y^{(n-1)/2}\}$ with the start symbol replaced by S'. The CFG G for $\{Y^n\}$ is the union of G' and the two rules $S \rightarrow YS''$ and $S'' \rightarrow S'S'$. This CFG has two more nonterminals than G'. Hence the number of nonterminals in G is $\le 2\lg((n-1)/2) + 2 \le 2\lg n$.

2) Add the the productions $Y \rightarrow a$ and $Y \rightarrow b$ to the CFG from Part 1.

3) Add the production $Y \rightarrow \epsilon$ to the CFG from Part 1.

4) Add the production $Y \rightarrow \epsilon$ to the CFG from Part 2.

5) Let G be the $O(\log n)$ sized CFG for $\{Y^n\}$ from Part 1. Let G' be the $O(1)$ sized CFG for Y^*. The CFG for the concatenation of $L(G)$ and $L(G')$ is an $O(\log n)$ sized CFG for $\{Y^{\ge n}\}$.

6) Add the productions $Y \rightarrow a$ and $Y \rightarrow b$ to the CFG from Part 5. ∎

Note 7.2 Lemma 7.1.1 follows from the first paragraph of the proof of Proposition 16 in the journal version of HH (Proposition 14 in the conference version). They used PDAs (which they call $\emptyset$-PDAs) where as we use CFGs and then convert them to PDAs.

Theorem 7.3 *For almost all n there exists a (natural) language A_n such that the following hold.*

1. $A_n, \overline{A_n} \in L(\mathrm{PDA})$.

2. *Any DPDA that recognizes* $\overline{A_n}$ *requires size* $\geq 2^{2^{\Omega(n)}}$.

3. *There is a PDA of size* $O(n)$ *that recognizes* A_n.

Proof: We show there is a language A_n such that (1) $A_n, \overline{A_n} \in L(\mathrm{PDA})$, (2) any PDA that recognizes $\overline{A_n}$ requires size $\geq 2^{\Omega(n)}$, (3) there is a PDA of size $O(\log n)$ that recognizes A_n. Rescaling this result yields the theorem.

Let $W_n = \{ww : |w| = n\}$. Let $A_n = \overline{W_n}$.

1) A_n is cofinite, so both A_n and $\overline{A_n}$ are in $L(\mathrm{PDA})$.

2) Filmus [2] showed that any CFG for W_n requires size $\geq 2^{\Omega(n)}$. Hence by Example 1.4.4 any PDA for $W_n = \overline{A_n}$ requires size $\geq 2^{\Omega(n)}$.

3) We present a CFG for A_n of size $O(\log n)$. By Example 1.4.5 this suffices to obtain a PDA of size $O(\log n)$. We will freely use that Lemma 7.1 yields CFG's of size $O(\log n)$ by Example 1.4.4.

Note that if $x \in A_n$ then either $|x| \leq 2n-1$, $|x| \geq 2n+1$, or there are two letters in x that are different and are exactly $n-1$ apart. These sets are not disjoint.

The CFG is the union of three CFGs. The first one generates all strings of length $\leq 2n-1$. By Lemma 7.1 there is such a CFG of size $O(\log n)$. The second one generates all strings of length $\geq 2n+1$. By Lemma 7.1 there is such a CFG of size $O(\log n)$.

The third one generates all strings of length $\geq 2n$ where there are two letters that are different and exactly $n-1$ apart (some of these strings are also generated by the second CFG). By Lemma 7.1 there is a CFG G' of size $O(\log n)$ that generates all strings of length $n-1$. Let S' be its start symbol. G' will be part of our CFG G.

Our CFG has start symbol S, all of the rules in G', and the following:

$S \to UaS'bU \quad | \quad UbS'aU$

$U \to aU \quad | \quad bU \quad | \quad \epsilon$

The union of the three CFG's clearly yields a CFG of size $O(\log n)$ for A_n. ∎

Note 7.4 Theorem 7.3 is implicit in Proposition 27 of HH. They use the language

$$\Sigma^* - \{w\$w\$w\$w : |w| = n\}.$$

We can now obtain a double exponential result about (DPDA,PDA).

Theorem 7.5 *For almost all n there exists a (natural) language A_n such that the following hold.*

1. *Any DPDA that recognizes A_n requires size $\geq 2^{2^{\Omega(n)}}$.*
2. *There is a PDA of size $O(n)$ that recognizes A_n.*

Proof: Let A_n be as in Theorem 7.3 (note that it is scaled). We already have that A_n has a PDA of size $O(n)$. We show that any DPDA for A_n is of size $\geq 2^{2^{\Omega(n)}}$. Let P be an DPDA for A_n. By Example 1.8.2 there is a DPDA P' for $\overline{A_n}$ of size $O(|P|)$. By Theorem 7.3 $|P'| \geq 2^{2^{\Omega(n))}}$, hence $|P| \geq 2^{2^{\Omega(n)}}$, ∎

Note 7.6 Theorem 7.5 is a special case of Corollary 30 of HH.

8 A Double-Exp For-Almost-All Result Via a Natural Language for (PDA,LBA)

We show that for almost all n there is a (natural) language A_n that has a small LBA but requires a large PDA.

Theorem 8.1 *For almost all n there exists a (natural) language A_n such that the following hold.*

1. *Any PDA that recognizes A_n requires size $\geq 2^{2^{\Omega(n)}}$.*
2. *There is an LBA of size $O(n)$ that recognizes A_n.*

Proof: We show there is a language A_n such that (1) any PDA for A_n requires size $\geq 2^{\Omega(n)}$ and (2) there is an LBA of size $O(\log n)$ for A_n. Rescaling this result yields the theorem. Let

$$A_n = \{w\$w : |w| = n\}.$$

1) Filmus [2] showed that any CFG for A_n requires size $\geq 2^{\Omega(n)}$. Hence, by Example 1.4.4, any PDA for A_n requires size $\geq 2^{\Omega(n)}$.

2) We present a CSG for W_n of size $O(\log n)$. By Example 1.4.7 this yields an LBA of size $O(\log n)$.

Here is the CSG for $\{w\$w : |w| = n\}$.

$S \to Y^n W$ (actually use the CFG from Lemma 7.1 of size $O(\log n)$ to achieve this)

$Y \to aA \quad | \quad bB$

$Aa \to aA$

$Ab \to bA$

$Ba \to aB$

$Bb \to bB$

$AW \to Wa$

$BW \to Wb$

$W \to \$$ ▮

9 A Ginormous For-Almost-All Result for (PDA,LBA)

Meyer and Fisher [11] say the following in their **Further Results** Section:

. . . *context-sensitive grammars may be arbitrarily more succinct than context-free grammars* . . .

The reference given was a paper of Meyer [10]. That paper only refers to Turing Machines. We exchanged emails with Meyer about this and he informed us that his techniques could be used to obtain the result that is Theorem 9.1 below. Rather than work through his proof we provide our

own. Our proof is likely similar to his; however, we use the closure of $L(\mathrm{LBA})$ under complementation [8, 15] which was not available to him at the time.

We assume that all LBAs are modified so that, on input x, if a branch does not terminate in time $2^{|x|^2}$ then that branch will halt and reject. Hence every branch either halts and accepts or halts and rejects.

Let $P_1, P_2, \ldots$ be a size-enumeration of all PDAs. We assume that P_e is of size $\geq e$. We also have a list $N_1, N_2, \ldots,$ of LBAs such that $L(N_i) = L(P_i)$ and (by the effective closure of $L(\mathrm{LBA})$ under complementation) $N_1', N_2', \ldots$ such that $L(N_i') = \overline{L(P_i)}$. Note that $N_1, N_2, \cdots$ *is not* a list of all LBAs.

The following will be key later: Let $x \in \Sigma^*$ and imagine running $N_i(x)$, for each path noting if it said Y or N, and then running $N_i'(x)$, and then noting if that path said Y or N. So each path ends up with a NN, NY, YN, or YY.

- $x \in L(P_i)$: some path says YN, some paths might say NN, but no path says NY or YY.
- $x \notin L(P_i)$: some path says NY, some path might says NN, but no path says YN or NN.

Note that N_i and N_i' run in $O(|x|)$ space.

Theorem 9.1 *Let $f \leq_{\mathrm{T}} HALT$. For almost all n there exists a finite set A_n such that the following hold.*

1. *Any PDA that recognizes A_n requires size $\geq f(n)$.*
2. *There is an LBA of size $O(n)$ that recognizes A_n.*

Proof: We construct the language A_n by describing an LBA for it (really an $\mathrm{NSPACE}(|x|)$ algorithm). The idea is that A_n will be diagonalized against all small PDAs. The algorithm will run in $O(|x|)$ space. We will comment on the constant in the $O(|x|)$ later.

Since $f \leq_{\mathrm{T}} HALT$, by Fact 2.1.6, there exists a computable g such that $(\forall n)[f(n) = \lim_{s\to\infty} g(n,s)]$. We can assume $g(n,s)$ can be computed in space $O(\log(n+s))$.

Fix n. We describe the algorithm for A_n. The set we construct will satisfy the following requirements:

For $1 \le i \le f(n)$ (which we do not know)

$R_i : A_n \neq L(P_i)$.

This is only a finite number of requirements; however, we do not know $f(n)$. We will get around this by approximating $f(n)$ via $g(n, s)$.

The set A_n will be a subset of a^*.

ALGORITHM for A_n

1. Input(a^s).
2. Compute $t = g(n, s)$.
3. Deterministically simulate A_n on the strings $\{\epsilon, a, a^2, \ldots, a^{\lg^* s}\}$. Do not store what the results are; however, store which requirements indexed $\le t$ are satisfied. If so many were satisfied that you can't store them in space $\le \log s$ then reject and halt.
4. If all of the requirements P_i as $1 \le i \le t$ are satisfied then reject and halt.
5. (Otherwise) Let i be the least elements of $\{1, \ldots, t\}$ such that R_i has not been seen to be satisfied. Run (nondeterministically) $N_i(x)$ and then $N_i'(x)$. Any path that yields NN outputs NO. There will be no paths that yields YY. Any path that yields NY output YES (this is diagonalization— a NY means that $x \notin L(P_i)$). Any path that yields YN output NO (this is diagonalization— a YN means that $x \in L(P_i)$). Requirements R_i is satisfied.

END OF ALGORITHM for A_n

By the definition of g there exists s_0 such that, for all $s, s' \ge s_0$, $g(n, s) = g(n, s_0) = f(n)$.

We show, by induction on i, that for all $i \le f(n)$, R_i is satisfied. Assume that for all $i' < i \le f(n)$, $R_{i'}$ is satisfied. Let $s_1 > s_0$ be the least s such that all $R_{i'}$ with $i' < i$ are satisfied after $A_n(a^s)$ is determined. Let $s_2 > s_1$ be the least s such that for all inputs $a^{\ge s}$ the algorithm deterministically

simulates A_{s_1} and hence notices that, for all $i' < i$, $R_{i'}$ is satisfied. If R_i is satisfied on some input in $a^{\le s_2}$ then we are done. Otherwise note that on input a^{s_2} the algorithm will notice that R_i is not satisfied and satisfy it.

How big is the LBA for A_n? The LBA only needs the parameter n and a constant number of instructions. Hence their is an LBA of size $O(\log n)$; however, we only need that there is an LBA of size $O(n)$.

We show that the algorithm for A_n is in $\mathrm{NSPACE}(O(|x|)$. Let $g_{\max} = \max\{g(n,s) : s \in \mathsf{N}\}$. Since $\lim_{s\to\infty} g(n,s)$ exists $g_{\max}$ is well defined. It depends on n but not on the input; hence $g_{\max}$ is a constant. The first four steps of the algorithm take $\le \lg^*(|x|) + g_{\max}$ space to execute. For large $|x|$ this is far less than $|x|$. Step 5 is the only nondeterministic step. Each branch is the result of running a branch of the $\mathrm{NSPACE}(O(|x|)$ machines $N_i(x)$ and $N_i'(x)$ where $1 \le i \le g_{\max}$. Hence there is a constant c such that for all x each branch of the computation takes $\le c|x|$ space. Therefore the algorithm for A_n is in $\mathrm{NSPACE}(O(|x|)$. ∎

10 Open Problems

We have pinned down the exact Turing degree of the bounding function for (DPDA,PDA) and (PDA,LBA). The exact Turing degree for the bounding functions for (DPDA,UCFG) and (UCFG,PDA) are open.

We have obtained natural languages that show the (1) bounding function for (DPDA,PDA) and (PDA,LBA), and (2) the c-bounding function for PDAs, are at least double exponential. It is open to find natural languages that show a larger lower bounds.

11 Acknowledgment

We thank Albert Meyer for conversations that helped clarify the history of Theorem 9.1. We thank Hermann Gruber, Albert Meyer, Leslie Valiant, and the referees for some references we had overlooked. We thank Yuval Filmus, Karthik Gopalan, Rebecca Kruskal, Sam Zbarsky, and

the referees for corrections and proofreading. We thank Yuval Filmus and one of the referees for pointing out an error in the original proof of Theorem 8.1 and supplying us with a way to fix the proof. We thank Keith Ellul, Bryan Krawetz, Jeffrey Shallit, and Ming-wei Wang whose paper [1] inspired this paper; and we thank Jeffrey Shallit who brought this paper to our attention.

References

[1] K. Ellul, B. Krawetz, J. Shallit, and M. Wang. Regular expressions: new results and open problems. *Journal of Automata, Languages, and Combinatorics*, 10(4):407–437, 2005.

[2] Y. Filmus. Lower bounds for context-free grammars. *Information Processing Letters*, 111(18):895–898, 2011.

[3] F. L. Gall. Power of tensors and fast matrix multiplication. In *Proceedings of the 2014 International Symposium on Symbolic and Algebraic Computation ISSAC2014*, Kobe, Japan, pages 269–303, 2014.

[4] H. Gruber, M. Holzer, and M. Kutrib. On measuring non-recursive trade-offs. In J. Dassow, G. Pighizzini, and B. Truthe, editors, *Proceedings of the 11th Annual workshop on Descrptional Complexity of Formal Systems (DCFS 2009)*, pages 141–150, 2009.

[5] D. Harel and T. Hirst. On the power of bounded concurrency ii: Pushdown automata. *Journal of the ACM*, 41(3):540–554, 1994. Earlier conference version in Colloquium on Trees and Algebra in Programming (CAAP) 1990. The journal version is behind a paywall and hence lost to humanity forever; however, the conference version is available on David Harel's website.

[6] J. Hartmanis. On the succinctness of different representations of languages. *SIAM Journal on Computing*, 9(1):114–120, 1980.

[7] L. Hay. On the recursion-theoretic complexity of relative succinctness of representations of languages. *Information and Computation*, 52(1):1–7, 1982.

[8] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, 1988. Prior version in *C*onf. on Structure in Complexity Theory, 1988.

[9] L. Lee. Fast contex-free grammar parsing requires fast boolean matrix multiplication. *Journal of the ACM*, 49(1):1–15, 2002.

[10] A. Meyer. Program size in restricted programming languages. *Information and Control*, 21(4):382–394, 1972.

[11] A. Meyer and M. Fischer. Economy of description by automata, grammars and formal systems. In *Proceedings of the 12th Annual symposium on Switching and Automta Theory*, pages 188–191, Washington, DC, 1971. IEEE.

[12] E. Schmidt. Succintness of descriptions of unambigous context-free languages. Technical Report Technical Report 76-277, Cornell University, Dept of Computer Science, April 1976. `http://ecommons.library.cornell.edu/handle/1813/7319`.

[13] J. Shoenfield. On degrees of unsolvability. *Annals of Mathematics*, 69(3):644–653, 1959.

[14] R. E. Stearns. A regularity test for pushdown machines. *Information and Control*, 11(2):323–340, 1967.

[15] R. Szelepcsenyi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 25(3):279–284, 1988.

[16] L. Valian. A note on the succintness of descriptions of deterministic languages. *Information and Control*, 32(2):139–145, 1976.

[17] L. Valiant. General context-free recognition in less than cubic time. *Journal of Computer and System Sciences*, 10(2):308–315, 1975.

[18] L. Valiant. Regularity and related problems for deterministic pushdown automata. *Journal of the ACM*, 22(1):1–10, 1975.